Trần Văn Hảo B20DCCN227

1. Mô tả bằng dạng Bảng và ngôn ngữ tự nhiên các chức năng tương ứng với các actor và phi chức năng của Hệ thống ecomSys.

Xác đinh các actor của hệ thống

a) Primary actor
   - **Manager**: là người quản lý và duy trì các thông tin quan trọng trong hệ thống ecomSys. Họ có chức năng Quản lý thông tin sản phẩm, thêm sửa xóa sản phẩm.
   - **Customer**: là những người dùng cuối cùng của hệ thống, truy cập trang web ecomSys để thực hiện các hoạt động mua sắm. Customer tìm kiếm, xem sản phẩm, thêm sản phẩm vào giỏ hàng, đặt hàng thực hiện thanh toán trực tuyến và quản lý tài khoản cá nhân. Họ tạo đơn đặt hàng, đánh giá sản phẩm, và cung cấp phản hồi về trải nghiệm mua sắm

b) Secondary actor
   - Hệ thống thanh toán: Giúp xử lý các giao dịch thanh toán, hỗ trợ khách hàng thanh toán qua ví điện tử Pay Pal
   - Hệ thống thông báo: gửi thông báo và cập nhật đến khách hàng và các bên liên quan về các sự kiện quan trọng và hoạt động trong hệ thống qua trang web

Xác đinh chức năng ứng với mỗi actor

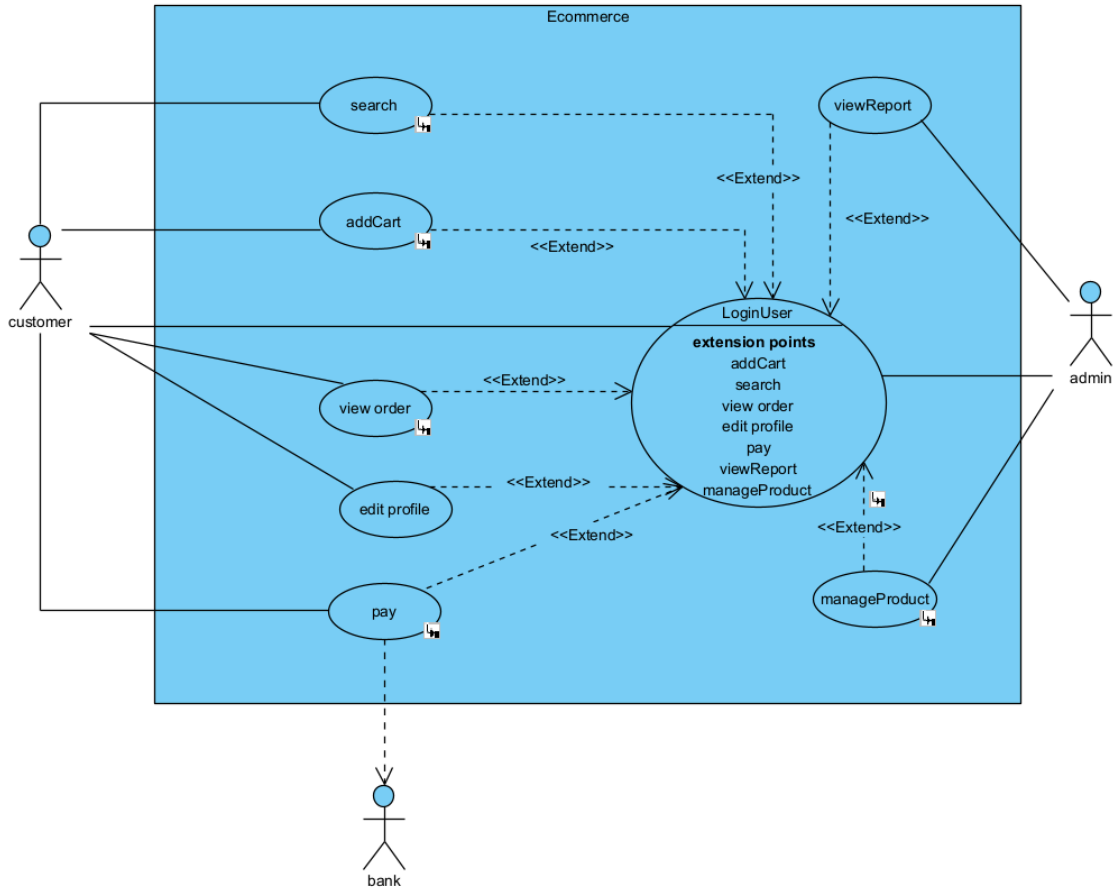
a) Manager
   - Quản lý sách: CRUD
   - Quản lý điện thoại: CRUD
   - Quản lý quần áo: CRUD
   - Quản lý danh mục sách: CRUD
   - Quản lý danh mục điện thoại: CRUD
   - Quản lý danh mục quần áo: CRUD
   - Quản lý thanh toán: CRUD
   - Xem thống kê: doanh thu các mặt hàng, lợi nhuận, doanh thu

b) Customer
   - Đăng ký thành viên
   - Đăng nhập hệ thống
   - Xem sản phẩm

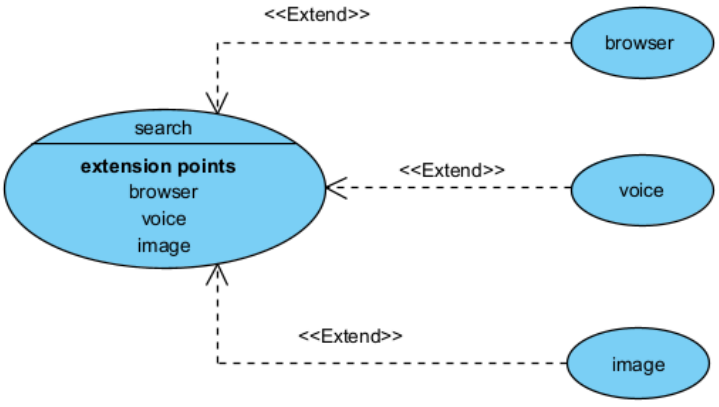
- Thêm sản phẩm vào giỏ hàng
- Đặt hàng (đặt qua giỏ hàng hoặc đặt 1 sản phẩm)
- Thanh toán đơn hàng
- Tìm kiếm sản phẩm
- Đánh giá sản phẩm

2. Vẽ Biểu đồ use case tổng quát và biểu đồ usecase chi tiết cho từng chức năng/dịch vụ

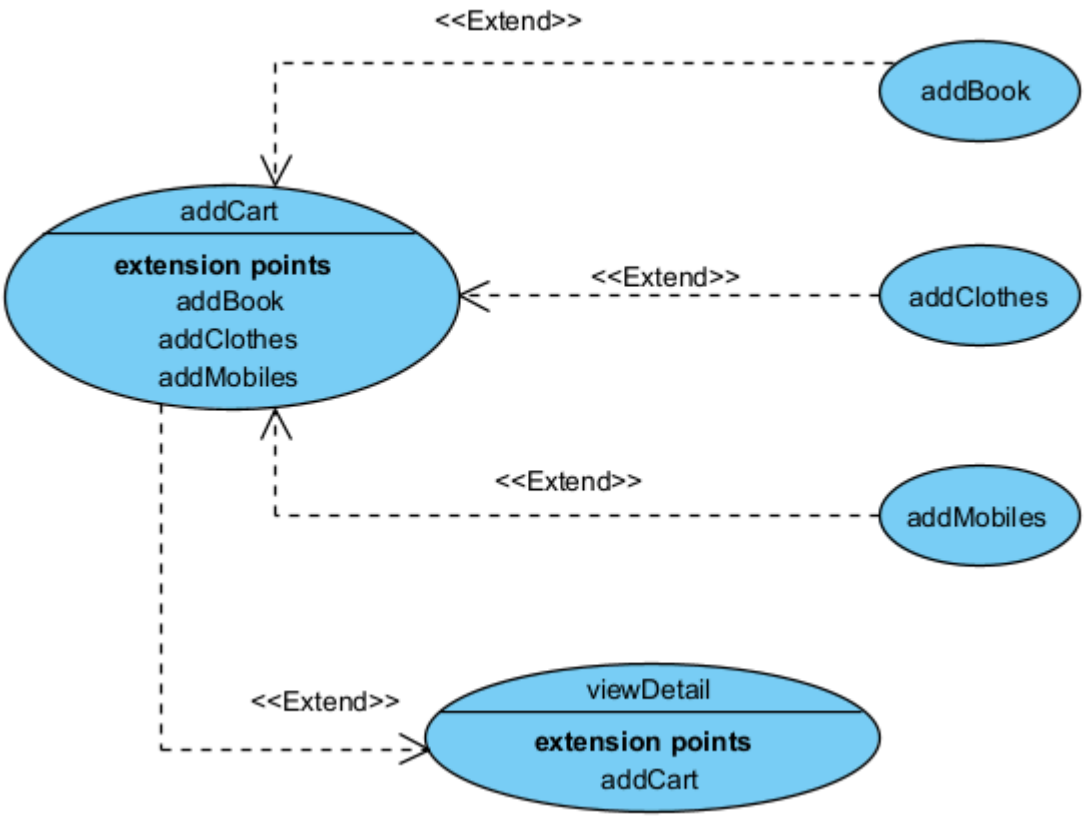
Biểu đồ usecase tổng quan



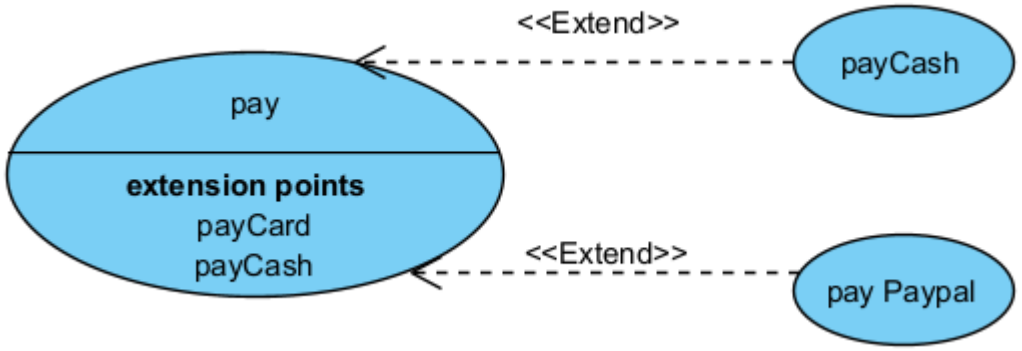
Chi tiết usecase search



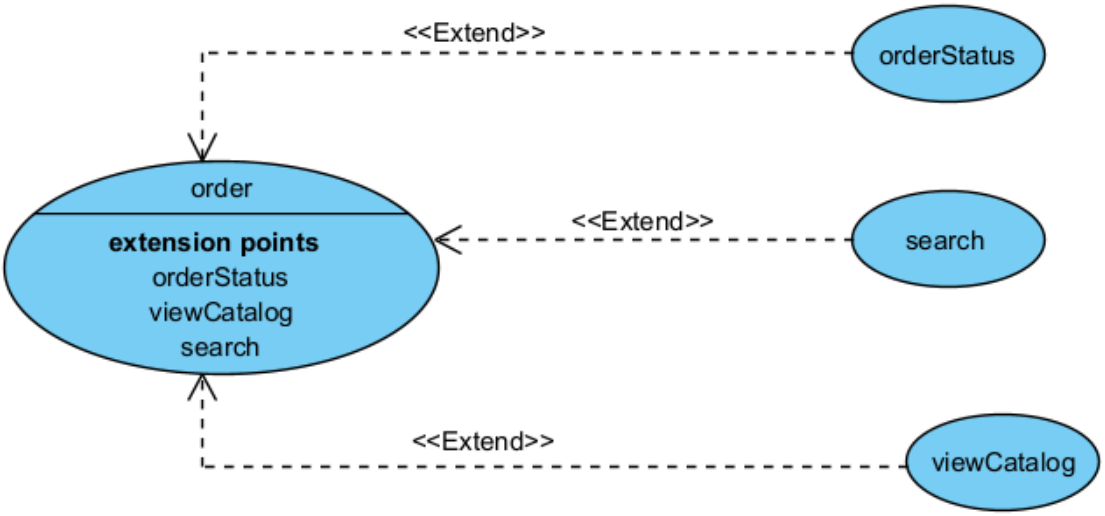
Chi tiết usease addCart



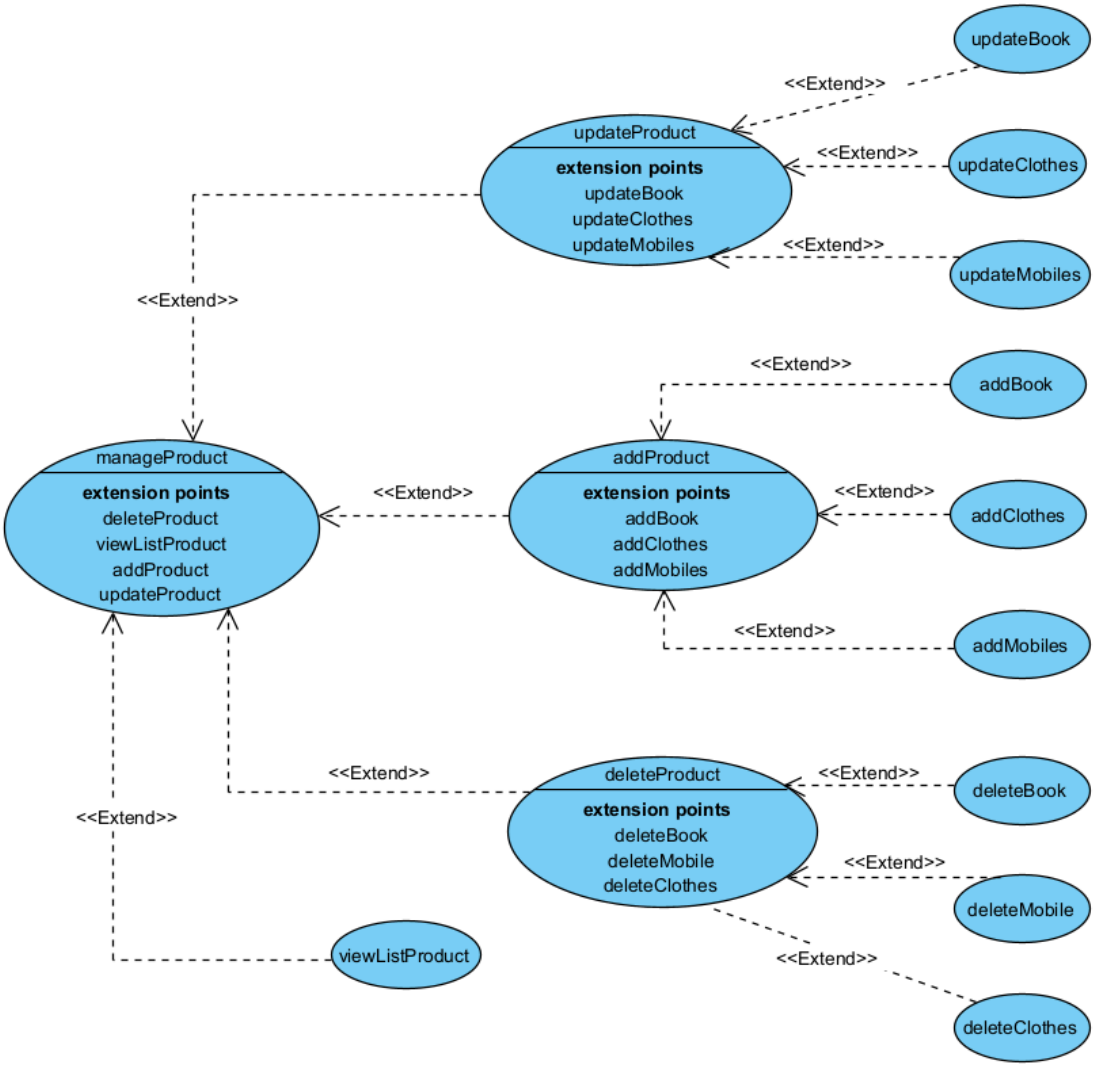
Chi tiết usecase pay



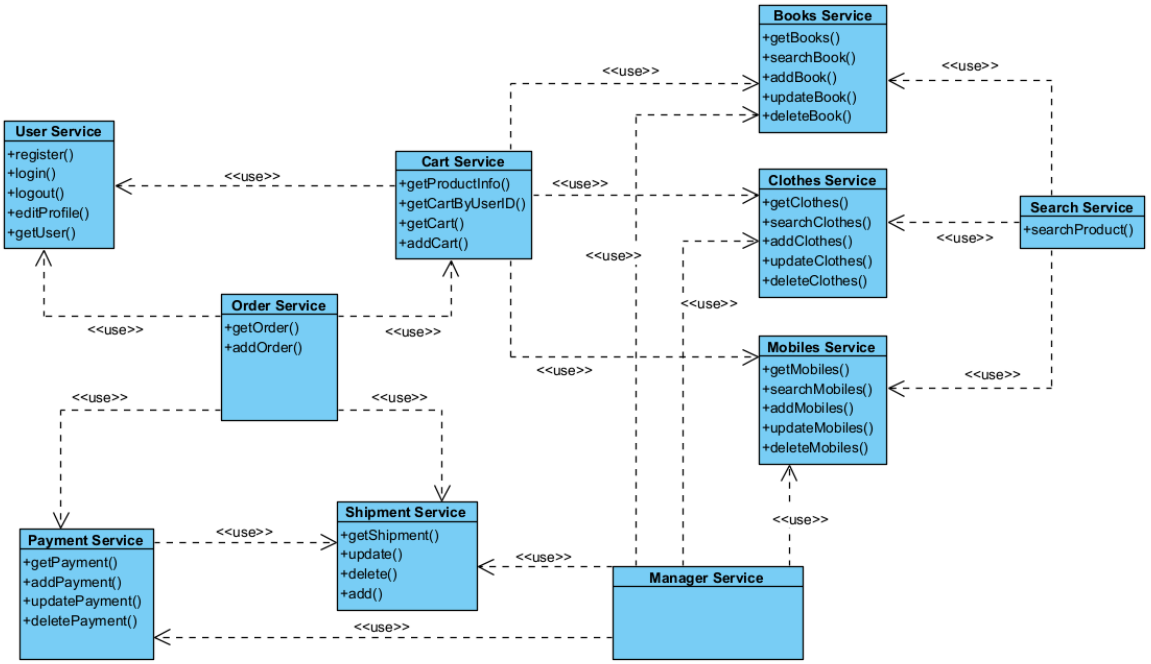
Chi tiết usecase order



Chi tiết usecase manage product

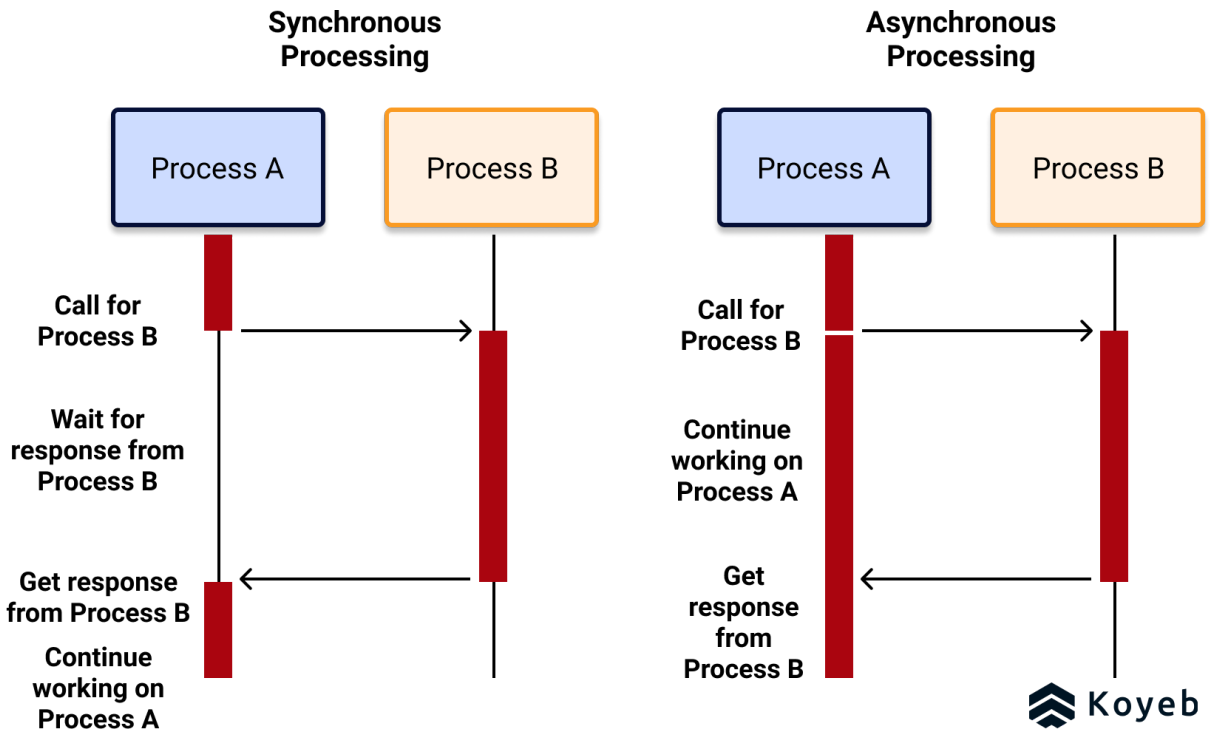


3. Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ (sử dụng quan hệ <<use>> trong UML)



4. Trình bày các dạng communication giữa các service với nhau (syschronous và asynchronous) với code và ví dụ. Cập nhật và bổ sung từ Bài tập 3.

Ví dụ minh họa syschronous và asynchronous



Giao tiếp đồng bộ (Synchronous communication):

Trong giao tiếp đồng bộ, dịch vụ gọi một dịch vụ khác và chờ đợi cho đến khi dịch vụ đó hoàn thành trước khi tiếp tục thực hiện các công việc khác. Đây là một ví dụ về giao tiếp đồng bộ sử dụng thư viện requests để gửi yêu cầu HTTP và chờ đợi phản hồi:

```
import requests

def get_user_details(user_id):
    response = requests.get(f'https://api.example.com/users/{user_id}')
    if response.status_code == 200:
        return response.json()
    return None

def process_user(user_id):
    user_details = get_user_details(user_id)
    if user_details:
        # Xử lý thông tin người dùng
        print(user_details)
    else:
        print('Không thể lấy thông tin người dùng')
```

Trong ví dụ trên, hàm get_user_details gửi một yêu cầu HTTP đồng bộ tới một API để lấy thông tin người dùng dựa trên user_id. Hàm process_user gọi

get_user_details và chờ đợi phản hồi trước khi tiếp tục xử lý thông tin người dùng.

Giao tiếp bất đồng bô (Asynchronous communication):

Trong giao tiếp bất đồng bộ, dịch vụ không chờ đợi cho đến khi dịch vụ khác hoàn thành. Thay vào đó, nó tiếp tục thực hiện các công việc khác trong khi chờ đợi phản hồi. Dưới đây là một ví dụ về giao tiếp bất đồng bộ sử dụng thư viện asyncio và aiohttp để gửi yêu cầu HTTP và xử lý phản hồi:

```
import asyncio
import aiohttp

async def get_user_details(user_id):
    async with aiohttp.ClientSession() as session:
        async with session.get(f'https://api.example.com/users/{user_id}') as response:
            if response.status == 200:
                return await response.json()
            return None

async def process_user(user_id):
    user_details = await get_user_details(user_id)
    if user_details:
        # Xử lý thông tin người dùng
        print(user_details)
    else:
        print('Không thể lấy thông tin người dùng')

loop = asyncio.get_event_loop()
loop.run_until_complete(process_user('123'))
```

Trong ví dụ trên, hàm get_user_details sử dụng aiohttp để gửi một yêu cầu HTTP bất đồng bộ để lấy thông tin người dùng. Hàm process_user sử dụng await để chờ đợi phản hồi từ get_user_details trong khi tiếp tục thực hiện các công việc khác. Cuối cùng, run_until_complete được sử dụng để chạy hàm process_user trong một vòng lặp asyncio.

5. Trình bày sử dụng các dạng communication giữa các service với code cho hệ ecomSys

Giao tiếp bất đồng bộ:

- Payment service: cần đợi người dùng xác nhận thanh toán và xác thực được người dùng mới bắt đầu thanh toán. Và cần đợi hệ thống xử lý

thanh toán xong mới có thể lấy được trạng thái thanh toán và update trạng thái thanh toán

- Search service: các api search book của service book, search clothes của service clothes, search mobile của service mobile có thể thực hiện bất đồng bộ sau khi nhận được từ khóa
- User service: chỉ cần đợi mỗi api trả về nên áp dụng bất đồng bộ
- Book service: thực hiện bất đồng bộ vì chỉ việc lấy 1 api duy nhất.
- Tương tự với Clothes service và Mobile Service

Giao tiếp đồng bộ:

- Order Service: Phải lấy được thông tin order trong getOrder() thì mới có thể lấy được các thông tin về sản phẩm trong Order
- Cart service: Trong phương thức thêm giỏ hàng cần phải đẩy dữ liệu lên, sau đó mới có thể tiến hàng lấy tất cả các sản phẩm trong cart hiện tại